Session 705

Apple Performance Tools

Robert Bowdidge
Developer Tools Group, Apple Computer

Improving Application Performance

- You've got great apps…
- …and you're shipping in July… right?
  - Is your application performing at its best?
  - How do you figure out what to fix?
- Need tools to expose app performance details

What Will You Learn Today?

- Apple has tools to help
  - Tools can identify possible causes
  - You'll see quick examples of tool use
- See framework talks for specific, concrete advice
- Explore tool use on your own

Causes of Poor Performance

- Excessive use of memory
- Executing too much code
- Waiting for other processes or devices
- Excessive drawing or redrawing

- Most problems cause excessive memory use
- Most memory use problems cause swapping

What Tools Exist?

Execution behavior	Heap use	System state
Sampler, sample	MallocDebug, ObjectAlloc, heap, leaks, malloc_history	QuartzDebug fs_usage, sc_usage, vmmap, top

What Tools Exist?

- System state: top, vmmap, QuartzDebug
- Execution behavior: Sampler, sample
- Heap use: MallocDebug, ObjectAlloc, heap, leaks, malloc_history
- All tools work on any binary without recompiling

Example Program: ThreadViewer

- Simple Cocoa application
- Shows activity on threads via timelines
- Realistic example, with real bugs
- What's wrong with ThreadViewer?
  - No obvious problems visible
  - Need baseline data to notice regressions

What's Happening on the System?

- First step: run top command line tool
- Provides details about system state
  - CPU, memory usage of tasks
  - Paging rate
- Most relevant: CPU Usage, swapping, RPRVT
  - Notice ThreadViewer never uses 100% of CPU time
  - RPRVT seems constant… or is it growing?
- top -d to see messages to Window Manager

Top: Check Memory Use Over Time

- How can you identify trends in memory use?
  - Call top repeatedly, save results to file
- Command line scripts can be your friend:
  - permit customized data collection
  - Run remotely, run without affecting system
  - Run even when window server won't respond

heap's Output

```
% heap Thread
Process 357: 2 zones
Zone CoreGraphicsDefaultZone_0x89f1e0: 844 nodes malloced
    for 89KB - Overall allocation 256KB - Largest unused:
    [0x8b749e-158KB]
Zone DefaultMallocZone_0x1491e0: 11904 nodes malloced for
    1938KB - Overall allocation 2192KB - Largest unused:
    [0x152806e-199KB]
All zones: 12748 nodes malloced - 2027KB
```

vmmap's Output

```
MALLOC_USED     139a000 [  16K] rw-/rwx SM=PRV
MALLOC_FREE     139e000 [  16K] rw-/rwx SM=PRV
MALLOC_USED     13a2000 [ 136K] rw-/rwx SM=PRV
                13c4000 [ 148K] rw-/rw- SM=SHM
STACK [3]       13ea000 [ 512K] rw-/rwx SM=PRV
                146b000 [   4K] rw-/rwx SM=COW
                146d000 [   4K] rw-/rwx SM=COW
                146f000 [   4K] rw-/rwx SM=COW
                1471000 [   4K] rw-/rwx SM=COW
                1473000 [   4K] rw-/rwx SM=COW
                1475000 [   4K] rw-/rwx SM=COW
```

Leaking Pages In task_threads

```
thread_act_array_t threadArrayFromMach;
ret = task_threads(task,&threadArray,&threadCount);
If (ret != NULL) {
  mach_error("task_threads",returnVal);
  return;
}
// query status of each thread
vm_deallocate(mach_task_self(),threadArray,
                    threadCount*sizeof(thread_t));
```

Causes of Excessive Memory Use

- Large data structures, scattered across memory
- Unneeded caching of data in memory
- Leaking unneeded memory

- Excessive memory use leads to more pages
- More pages lead to more swapping

Finding Causes of Memory Use

- MallocDebug: track source of current allocations
  - Relates allocations to call graphs
  - Leak detection
  - Pointer problems
- ObjectAlloc: how many objects exist?
  - Shows number of objects of each class
  - Shows changes over time

MallocDebug: Where Is Memory Allocated?

- Shows all allocations, grouped by allocation site
  - Browse call graph to find uses at site
  - Examine contents of memory
  - Identify calls responsible for large allocations
- Only shows currently allocated objects

MallocDebug: Are We Leaking Any Memory?

- What memory are we allocating, then forgetting?
  - Leaks waste memory, fatal for long-running apps
  - Force scattering of memory that's used
- MallocDebug can detect unreferenced blocks
  - Searches all memory for pointers to blocks
  - Unreferenced blocks must be leaked
  - Only root of leaked structure shown!

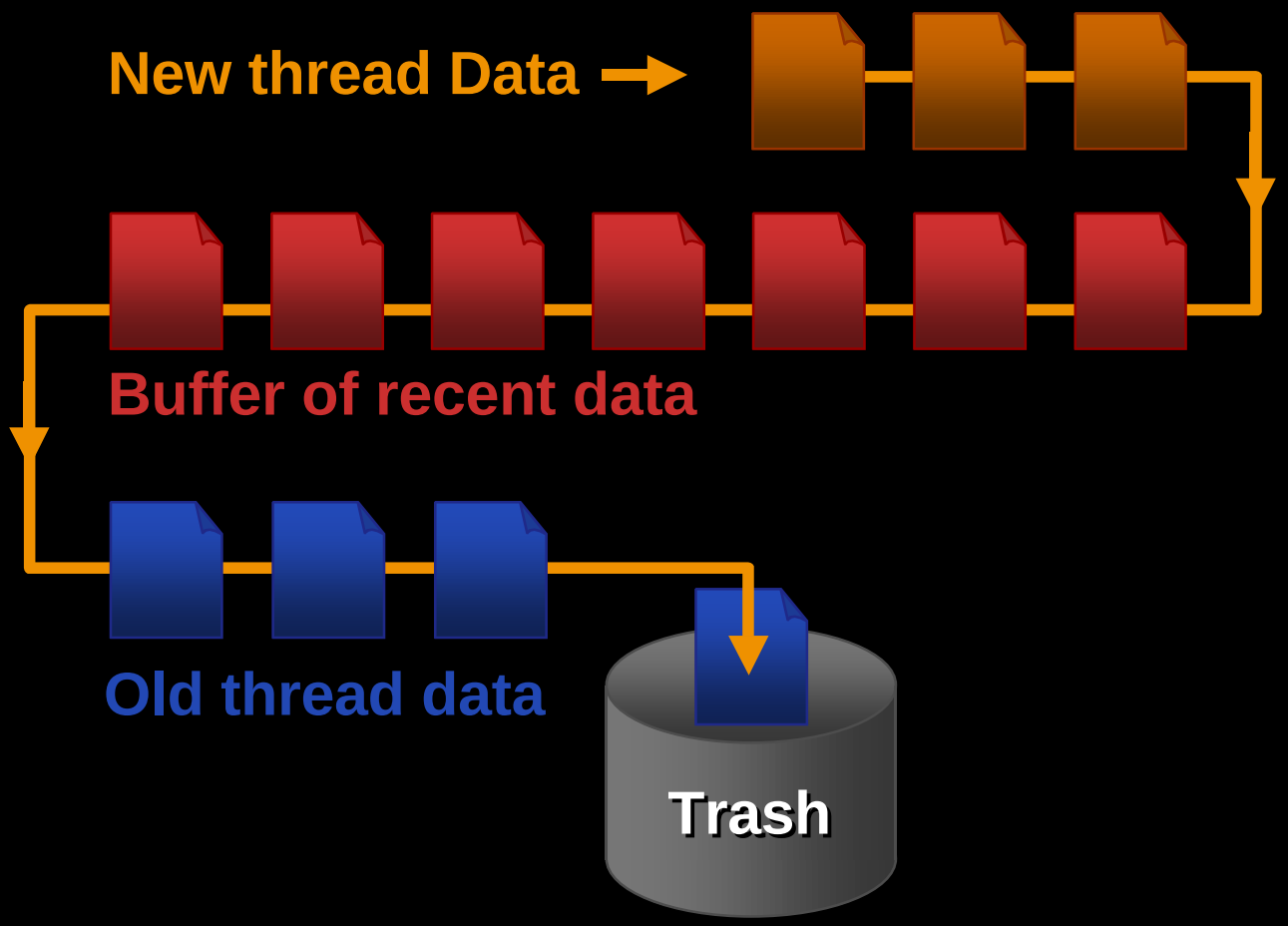
MallocDebug: Finding Pointer Bugs

- Some memory bugs can be subtle, intermittent
  - Referencing freed memory causes values to change
  - Overrunning/underrunning buffers trash memory
- MallocDebug "helps" detect such problems
  - Encourages badly behaved program to crash
  - Overwrites freed memory, adds guard words
  - Assorted other warning messages to console

ObjectAlloc: Analyzing Object Use

- ObjectAlloc shows how many objects exist
  - Current, max, total created over life of program
  - Shows totals, bar graph, backtraces
- Great for summary, noting trends
  - Did you expect 100 FooBars?
- Works for CoreFoundation, ObjC objects
  - Backtraces for Objective C retains and releases
  - Also shows malloc blocks

Are We Correctly Freeing Samples?



CPU Usage: What Are The Problems?

- Cause of excessive CPU time
  - Executing unnecessary code
  - Expensive algorithm
  - Unexpected calls to expensive code
  - Checking for events by polling, not blocking
- General approach: find most costly routines
  - Improvements will make largest difference

CPU Usage: Where Are Hot Spots?

- To improve speed, you *need* to find costly calls
- Sampler finds what code is executing
  - Stops program at intervals, checks call stacks
  - Shows only functions seen at sample time
  - Presents call graph with sample counts
- If you need better detail, check out gprof

Are We Using The Disk Badly?

- What's happening at startup to keep us slow?
- Maybe we're doing too many disk accesses
  - Accesses at wrong time gives slow appearance
- Use **fs_usage** to watch disk operations
  - Must be run as root or with **sudo**
- Use Sampler to see where we called routines

Thread Viewer's Log Code

- When gathering new thread data:

```
_logFile = fopen("/tmp/threadViewerLog","w+");
s = [currentSamples description];
fprintf(_logFile,"%s",[s cString]);
fclose(_logFile);
```

Are We Drawing Too Much?

- Drawing directly affects performance
  - Drawing consumes CPU time, memory
  - Idle windows consume memory
  - Drawing eventually done by window server
- Use QuartzDebug to understand drawing
  - Immediately perceive redundant drawing
  - Learn about offscreen windows

Other Tools to Check Out Yourself

- sc_usage: which system calls are used?
- Heap memory use
  - heap, leaks, malloc_history
- Process state: sample, vmmap

Hints About Tool Use

- All tools work on any binary
  - No need to instrument or recompile
- CFM binaries generated by CodeWarrior
  - Use inline traceback tables for symbol info
  - Set macro to use system's malloc library
  - ObjectAlloc: need to explicitly run LaunchCFMApp

Conclusion

- Tune your application to make best impression
- Aim to reduce memory use… in all ways
- Compare performance between builds
- Watch Window Manager and other servers
- Create great applications for Mac OS X

How To Learn More About Tools

- Part of Mac OS X Developer Tools CD
- Documentation for applications
  - Graphical tools: documentation in application
  - Command-line tools: man pages
  - /Developer/Documentation/ReleaseNotes
- Inside Mac OS X books
  - Inside Mac OS X: Performance
  - Inside Mac OS X: System Overview

Information Resources

Information pages for all Mac OS X tools
http://developer.apple.com/tools/

Project Builder web page
http://developer.apple.com/tools/projectbuilder/

Apple's mailing list master page
http://www.lists.apple.com/

Mailing lists:

projectbuilder-users	**projectbuilder-users**
cocoa-dev	**cocoa-dev**
carbon-dev	**carbon-dev**

Roadmap

121 Carbon Performance Tuning: How to tune a Carbon App	Hall 2 **Thurs., 2:00pm**
122 Using Cocoa: Techniques for Cocoa development	Hall 2 **Thurs., 3:30pm**
706 Debugging on Mac OS X: Using Project Builder and gdb	Hall 2 **Fri., 9:00am**
FF119 Feedback Forum: **Apple Developer Tools**	Room J1 **Fri., 3:30pm**

Who to Contact

Godfrey DiGiorgi
Technology Manager, Developer Tools and Cocoa
Apple Worldwide Developer Relations
ramarren@apple.com

Developer tools group e-mail:
macosx-tools-feedback@group.apple.com

Q&A

Godfrey DiGiorgi
Technology Manager for Development Tools
ramarren@apple.com

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001